

# 컴파일러 과제4

## 보고서

20181755 이건희

2024년 10월



## 1. 개발 환경

기존과 동일한 wsl2 상의 Ubuntu 22 LTS, gcc 11, gdb 12이다.

```
kh@ThinkPad-T16g2: ~  
kh@ThinkPad-T16g2:~$ flex --version  
flex 2.6.4  
kh@ThinkPad-T16g2:~$ bison --version  
bison (GNU Bison) 3.8.2  
Written by Robert Corbett and Richard Stallman.  
  
Copyright (C) 2021 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
kh@ThinkPad-T16g2:~$
```

lex로 flex, yacc로 bison을 사용하였다. 이들은 전부 Ubuntu Repository에서 제공한다.

## 2. 단계별 설명

### 2-1. 필요한 파일, 명세서 작성

#### 2-1-1. type.h

교재의 부록 1을 그대로 사용하였다.

type.h는 부록 1에 수록 해두었다.

#### 2-1-2. yacc specification (yacc.y)

```
yacc.y:293.1-22: warning: nonterminal useless in grammar: co  
293 | conditional_expression  
    | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
yacc.y: warning: 35 shift/reduce conflicts [-Wconflicts-sr]  
yacc.y: warning: 55 reduce/reduce conflicts [-Wconflicts-rr]  
yacc.y: note: rerun with option '-Wcounterexamples' to gener
```

처음 만들고 나서 yacc를 실행하니 shift/reduce 에러가 35개, reduce/reduce 에러가 55개가 발생해서 이걸 잡는데 꽤나 고생을 했다.

3주차 강의교안과 교재의 부록 3을 참고하였다. 10/8(화) 수업 중 교수님이 본 과제는 yylval()을 아직 쓸 단계가 아니라 하셔서, 이와 관련된 부분은 넣지 않았다.

yyerror()가 교재의 부록 3과 다르다. 보고서의 마지막에서도 언급하겠지만, 부록 3의 yyerror()는 형태를 보아하니 어디서 어떤 에러가 발생했는지까지 알려줄 것 같이 생겼다. 그러나 LMS 상 과제 4의 공지에서 yyerror()는 단순히 msg 스트링 하나 출력하는 것으로 두라 했다. 일단은 LMS의 공지를 지키게끔 완성했고, 이렇게 되면 필요가 없어지는 syntax\_err, line\_no, yytext와 같은 변수들도 전부 넣지 않았다.

```

401
402 int main()
403 {
404     yyparse();
405     printf("end");
406
407     return 0;
408 }

```

의도한 방향으로 진행한다면 yacc.y가 완성되는 양상은 (올바르게 진행했다면) 하나로 정해져 있다. yacc.y(y.tab.c)에 들어갈 메인 함수 하나만 올리겠다.

yacc.y는 부록 2에 수록 해두었다.

### 2-1-3. lex specification (lex.l)

3주차 강의교안의 내용을 그대로 사용하였다.

lex.l은 부록 3에 수록 해두었다.

### 2-2. 수정

```

CMakeLists.txt  hw4.sh x
1  #!/bin/bash
2  yacc -d yacc.y
3  lex lex.l
4  cc y.tab.c lex.yy.c

```

(일련의 작업을 hw4.sh라는 간단한 shell script로 묶음)

```

kh@ThinkPad-T16g2: ~/com  +  v
kh@ThinkPad-T16g2:~/compiler/hw4$ ./hw4.sh
yacc.y: warning: 1 shift/reduce conflict [-Wconflicts-sr]
yacc.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
y.tab.c: In function 'yyparse':
y.tab.c:1513:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
1513 |         yychar = yylex ();
      |
kh@ThinkPad-T16g2:~/compiler/hw4$

```

컴파일 시에 오류는 없었으나 한 가지 경고가 발생했다.

```

Debug  parser  T  D  :
CMakeLists.txt  hw4.sh  lex.yy.c x
706
707 extern int yylex (void);
708
709 #define YY_DECL int yylex (void)
710 #endif /* !YY_DECL */
711
712 /* Code executed at the beginning of each rule, after y
713    * have been set up.
714    */
715 #ifndef YY_USER_ACTION

```

yylex는 lex.yy.c에 정의되어 있는데, 이를 y.tab.c에서 불러오지 못해서 경고가 발생하는 것이다.

```
yacc.y x
⚠ 명시적으로 이 파일은 평문으로 재할당
10 int yyerror(char *s);
11 int main();
12 extern int yylex();
13 %}
14
```

yacc.y에 한 줄 추가해주면, 이 내용은 그대로 y.tab.c에 들어간다.

## 2-3. yacc & lex 작동

```
kh@ThinkPad-T16g2: ~/com  × + ▾
kh@ThinkPad-T16g2:~/compiler/hw4$ yacc -d yacc.y
yacc.y: warning: 1 shift/reduce conflict [-Wconflicts-sr]
yacc.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
kh@ThinkPad-T16g2:~/compiler/hw4$ yacc -d yacc.y -Wcounterexamples
yacc.y: warning: 1 shift/reduce conflict [-Wconflicts-sr]
yacc.y: warning: shift/reduce conflict on token ELSE_SYM [-Wcounterexamples]
Example: IF_SYM LP expression RP IF_SYM LP expression RP statement • ELSE_SYM statement
Shift derivation
  selection_statement
    ↳ 92: IF_SYM LP expression RP statement
      ↳ 84: selection_statement
        ↳ 93: IF_SYM LP expression RP statement • ELSE_SYM statement
Reduce derivation
  selection_statement
    ↳ 93: IF_SYM LP expression RP statement
      ↳ 84: selection_statement
        ↳ 92: IF_SYM LP expression RP statement •
kh@ThinkPad-T16g2:~/compiler/hw4$ lex lex.l
kh@ThinkPad-T16g2:~/compiler/hw4$
```

yacc에서 shift/reduce conflict가 한 개 발생하는 것을 확인했다. bison에서는 conflict가 발생하는 상황을 만들기 위한 반례를 대입하는 기능을 제공한다. 이 기능으로 유일하게 발생한 conflict가 교수님이 수업중에 설명하신 if문의 ambiguous 때문임을 알 수 있었다. yacc는 y.tab.h, y.tab.c를 성공적으로 생성하였다. lex도 lex.yy.c를 성공적으로 생성하였다.

## 2-4. 간편한 테스트를 위한 shell script (hw4.sh)

```
kh@ThinkPad-T16g2: ~/com X + v
kh@ThinkPad-T16g2:~/compiler/hw4$ cat test.c
int main(){
    int a = 1;
    float b = 2

    return 0;
}
kh@ThinkPad-T16g2:~/compiler/hw4$ ./a.out < test.c
syntax error
kh@ThinkPad-T16g2:~/compiler/hw4$
```

(프로그램이 test.c 세 번째 줄의 마지막에 세미콜론이 없음을 알아냄)

사진과 같이 하나하나 입력해서 테스트를 하는 것은 너무나 번거로운 일이기에 또 다른 프로그램을 작성하여 yyparse()를 불러올 수도 있겠으나, 이보다도 더 간단한 shell script를 사용하겠다.

```
kh@ThinkPad-T16g2:~/compiler/hw4$ tree -I cmake-build-debug/
├── CMakeLists.txt
├── a.out
├── hw4.sh
├── hw4_20181755_이건희.hwp
├── hw4_20181755_이건희.pdf
├── lex.l
├── lex.yy.c
├── testdir
│   ├── 1.c
│   ├── 2.c
│   ├── 3.c
│   ├── 4.c
│   ├── 5.c
│   ├── 6.c
│   ├── 7.c
│   └── 8.c
├── type.h
├── y.tab.c
├── y.tab.h
└── yacc.y

1 directory, 19 files
kh@ThinkPad-T16g2:~/compiler/hw4$
```

과제 디렉터리의 구조는 위와 같다.

```
Debug v parser
hw4.sh x
1  #!/bin/bash
2  yacc -d yacc.y
3  lex lex.l
4  cc y.tab.c lex.yy.c
5
6  DIR="./testdir"
7  for file in "$DIR"/*.c
8  do
9      echo ""
10     echo "Parsing $file..."
11     ./a.out < "$file"
12 done
```

덧붙혀진 hw4.sh는 testdir의 모든 파일을 ./a.out에 pipe로 전달한다.

hw4.sh는 부록 4에 수록 해두었다.

### 3. 작동 확인

```
kh@ThinkPad-T16g2:~/compiler/hw4$ ./hw4.sh
yacc.y: warning: 1 shift/reduce conflict [-Wconflicts-sr]
yacc.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples

Parsing ./testdir/1.c...
Success

Parsing ./testdir/2.c...
syntax error

Parsing ./testdir/3.c...
Success

Parsing ./testdir/4.c...
Success

Parsing ./testdir/5.c...
syntax error

Parsing ./testdir/6.c...
Success

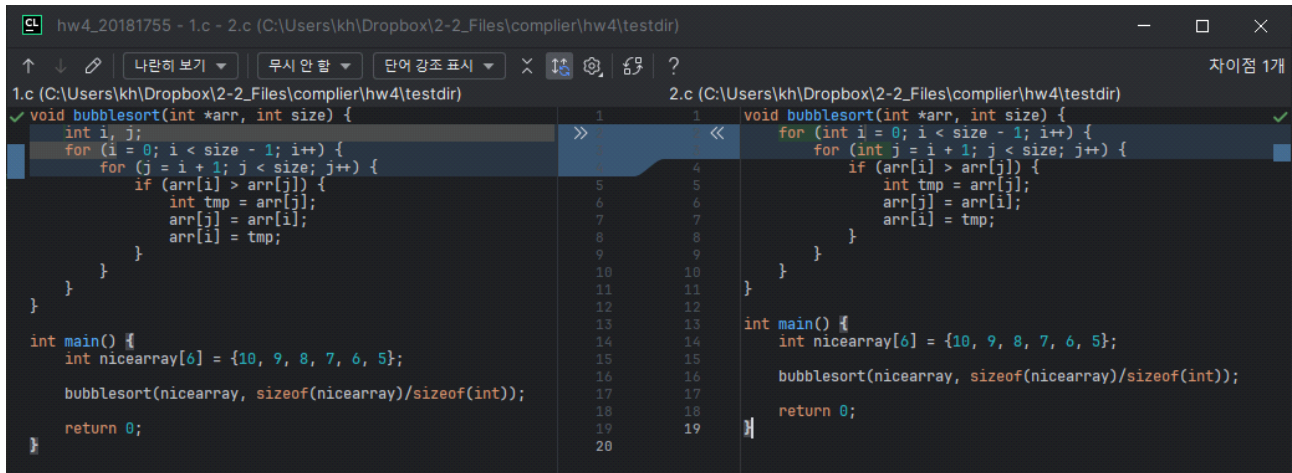
Parsing ./testdir/7.c...
Success

Parsing ./testdir/8.c...
syntax error
kh@ThinkPad-T16g2:~/compiler/hw4$
```

개인적으로 흥미가 있어서 Kernighan의 The C Programming Language, 2<sup>nd</sup> Edition을 읽고 C언어의 표준이 어떻게 바뀌어왔는지 찾아본 적이 있는데, 이 책은 C99 이전의 C Standard(K&R C, ANSI C)를 바탕으로 작성되었다. 강의 초반에 교수님이 알고계신 C Standard가 조금 옛날 기준이라는 말이 생각이 나서 테스트는 이에 중점을 두고 해보고자 한다. 이전의 과제에서도 그랬듯 잘 작동되는 상황보다 안되는 상황을 테스트 해보는 것만이 의미가 있다 생각한다. 이번에도 테스트의 개수는 자체는 적지만, 심사숙고 하여 정말 의미가 있는 코드들만 담았다.

테스트 파일들은 부록 5에 수록 해두었다.
-------------------------

### 3-1. 1.c vs 2.c

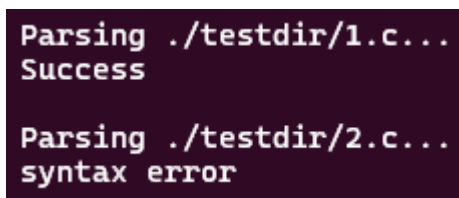


```
1.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir)
void bubblesort(int *arr, int size) {
    int i, j;
    for (i = 0; i < size - 1; i++) {
        for (j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                int tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;
            }
        }
    }
}

int main() {
    int nicearray[6] = {10, 9, 8, 7, 6, 5};
    bubblesort(nicearray, sizeof(nicearray)/sizeof(int));
    return 0;
}

2.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir)
void bubblesort(int *arr, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                int tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;
            }
        }
    }
}

int main() {
    int nicearray[6] = {10, 9, 8, 7, 6, 5};
    bubblesort(nicearray, sizeof(nicearray)/sizeof(int));
    return 0;
}
```

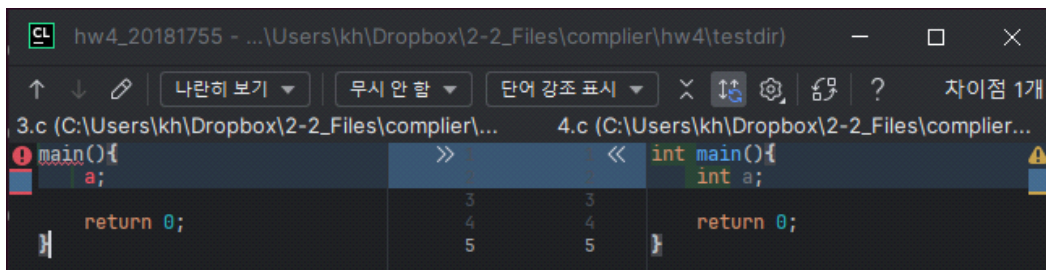


```
Parsing ./testdir/1.c...
Success

Parsing ./testdir/2.c...
syntax error
```

1과 2는 모두 버블소트를 하는 코드이다. for문에서 2와 같은 스타일로 코드를 적는 사람이 더 많으리라 생각된다. C99 이전의 C에서는 for문 내에서 반복자를 선언할 수가 없다. 따라서 본 프로그램은 2에서 syntax error가 발생했다 한다.

### 3-2. 3.c vs 4.c

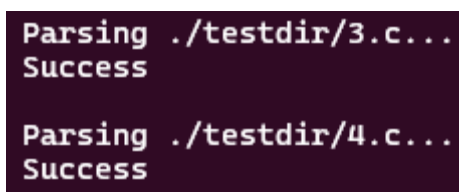


```
3.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir)
main() {
    a;

    return 0;
}

4.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir)
int main() {
    int a;

    return 0;
}
```



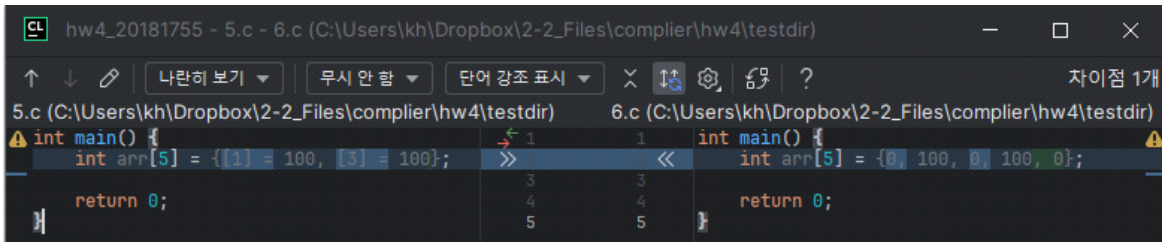
```
Parsing ./testdir/3.c...
Success

Parsing ./testdir/4.c...
Success
```

3은 ide에서 친절하게 경고까지 띄워준다. 일반적으로 알고 있는 C코드는 오른쪽의 형태일 것이다. 그러나 C99 이전의 C에서는 3의 스타일도 문법적으로 맞다. C99 이전의 C에서는 형식지정자가 없는 선언에 대해 묵시적으로 int 형식을 지정하였으나 C99부터는 모든 선언에 명시적으로 형식지정자를 지정 해주어야 한다. 한 3주차에 교수님이 이 내용을 질문하여 많은 학생들이 말문이 막혔었던 기억이 있다.



### 3-3. 5.c vs 6.c



```
5.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir) 6.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir)
int main() {
  int arr[5] = {[1] = 100, [3] = 100};
  return 0;
}

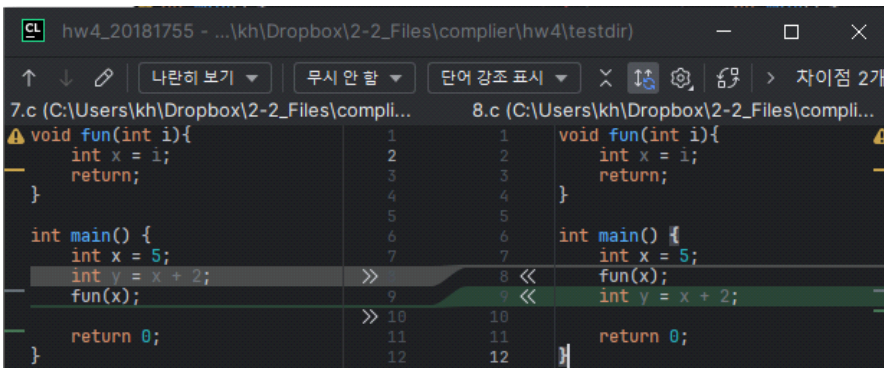
1 1 int main() {
  3 3 int arr[5] = {0, 100, 0, 100, 0};
  4 4 return 0;
  5 5 }
```

```
Parsing ./testdir/5.c...
syntax error

Parsing ./testdir/6.c...
Success
```

C99 이전의 C에서는 배열의 원하는 변수만 초기화가 불가능하다. 따라서 프로그램은 5에서 syntax error가 발생했다 한다. 지정된 초기화는 C99에서 생긴 기능이다.

### 3-4. 7.c vs 8.c



```
7.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir) 8.c (C:\Users\kh\Dropbox\2-2_Files\complier\hw4\testdir)
void fun(int i){
  int x = i;
  return;
}

int main() {
  int x = 5;
  int y = x + 2;
  fun(x);
  return 0;
}

1 1 void fun(int i){
  2 2 int x = i;
  3 3 return;
  4 4 }
  5 5
  6 6 int main() {
  7 7 int x = 5;
  8 8 fun(x);
  9 9 int y = x + 2;
 10 10
 11 11 return 0;
 12 12 }
```

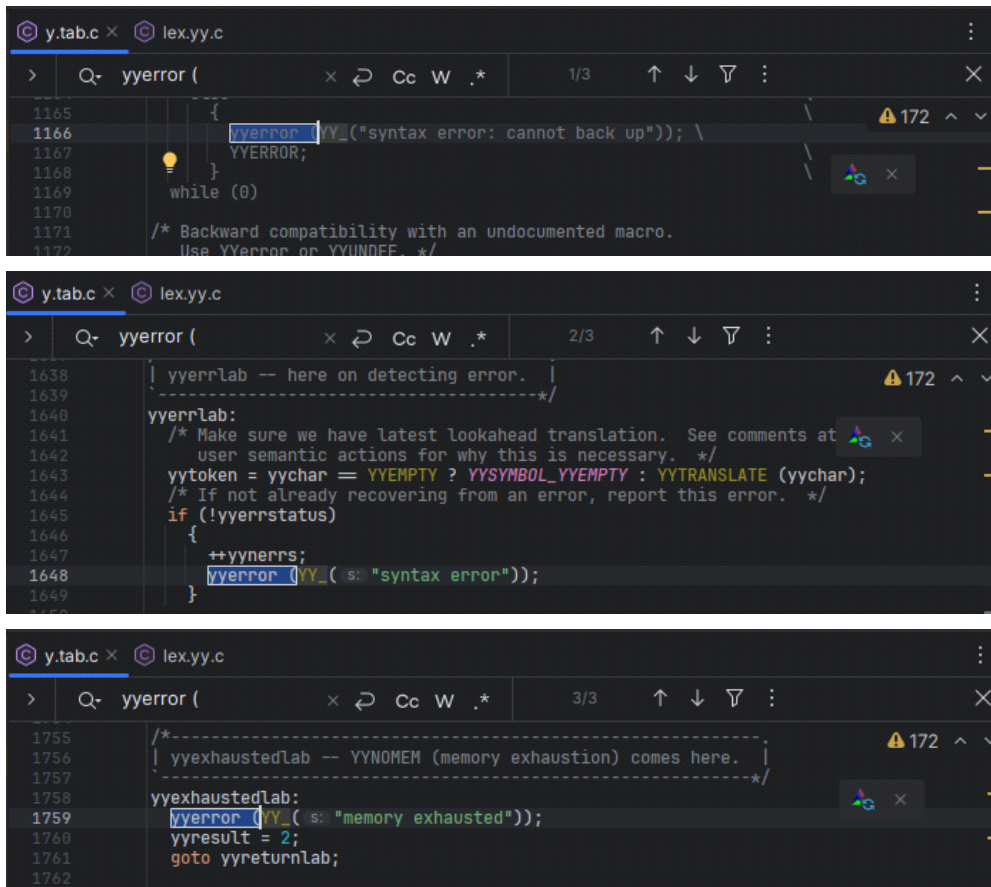
```
Parsing ./testdir/7.c...
Success

Parsing ./testdir/8.c...
syntax error
```

C99 이전의 C에서는 모든 변수는 실행문 이전에 선언 되어야 한다. 따라서 프로그램은 8에서 syntax error가 발생했다 한다. C99부터 선언과 코드의 순서가 자유로워졌다.

#### 4. 한계

2-1-2에서 yacc.y를 작성할 때도 똑같이 말한 내용이다. 우리가 일반적으로 사용하고 있는 컴파일러는 어디에서 어떤 에러가 발생했는지에 대한 정보를 준다. 하지만 본 프로그램에서는 이를 주지 않는데, yyerror()를 단순히 만들라는 LMS의 공지를 따랐기 때문이다.



테스트를 돌리면서 맞지 않는 코드에서는 단순히 “syntax error”만 나왔다. 에러를 처리하는 유일한 함수인 yyerror()가 어떻게 사용되는지 살펴보니 yacc.y로부터 만들어진 y.tab.c에서 세 번 등장하는데 오류의 종류 불문 전부 뭉통그려서 처리하도록 되어있다. 추후에 무조건 개선이 필요해 보인다.

#### 5. 결론

사실 과제 1이 Recursive-Descent 파싱 과정을 익히기 위한 가벼운 과제인 것 처럼, 본 과제 역시 yacc, lex의 사용법을 익히기 위한 가벼운 과제로 보여지고 실제로도 간단하게 완성이 가능했다. 비록 간단한 과제라지만 배끼지 않고(배길만 한 친구도 없고) 스스로의 힘으로 해결하였다는 점을 강조하고 싶고, 이렇기에 이후에 더 나은 파서, 컴파일러 역시나 스스로의 힘으로 만들 수 있으리라 생각한다.

## 부록 1. type.h

```
#define NIL 0
```

```
typedef enum {  
    FALSE, TRUE  
} BOOLEAN;
```

```
typedef enum e_node_name {  
    N_NULL,  
    N_PROGRAM, N_EXP_IDENT,  
    N_EXP_INT_CONST,  
    N_EXP_FLOAT_CONST,  
    N_EXP_CHAR_CONST,  
    N_EXP_STRING_LITERAL,  
    N_EXP_ARRAY,  
    N_EXP_FUNCTIONCALL,  
    N_EXP_STRUCT,  
    N_EXP_ARROW,  
    N_EXP_POST_INC,  
    N_EXP_POST_DEC,  
    N_EXP_PRE_INC,  
    N_EXP_PRE_DEC,  
    N_EXP_AMP,  
    N_EXP_STAR,  
    N_EXP_NOT,  
    N_EXP_PLUS,  
    N_EXP_MINUS,  
    N_EXP_SIZE_EXP,  
    N_EXP_SIZE_TYPE,  
    N_EXP_CAST,  
    N_EXP_MUL,  
    N_EXP_DIV,  
    N_EXP_MOD,  
    N_EXP_ADD,  
    N_EXP_SUB,  
    N_EXP_LSS,  
    N_EXP_GTR,  
    N_EXP_LEQ,  
    N_EXP_GEQ,  
    N_EXP_NEQ,  
    N_EXP_EQL,  
    N_EXP_AND,  
    N_EXP_OR,  
    N_EXP_ASSIGN,  
    N_ARG_LIST,  
    N_ARG_LIST_NIL,  
    N_STMT_LABEL_CASE,  
    N_STMT_LABEL_DEFAULT,  
    N_STMT_COMPOUND,  
    N_STMT_EMPTY,  
    N_STMT_EXPRESSION,  
    N_STMT_IF,  
    N_STMT_IF_ELSE,  
    N_STMT_SWITCH,  
    N_STMT_WHILE,  
    N_STMT_DO,  
    N_STMT_FOR,  
    N_STMT_RETURN,  
    N_STMT_CONTINUE,  
    N_STMT_BREAK,  
    N_FOR_EXP,  
    N_STMT_LIST,  
    N_STMT_LIST_NIL,  
    N_INIT_LIST,  
    N_INIT_LIST_ONE,  
    N_INIT_LIST_NIL  
} NODE_NAME;
```

```
typedef enum {  
    T_NULL_ENUM,  
    T_ARRAY,  
    T_STRUCT,  
    T_UNION,  
    T_FUNC,  
    T_POINTER,  
    T_VOID  
} T_KIND;
```

```
typedef enum {  
    Q_NULL,  
    Q_CONST,  
    Q_VOLATILE  
} Q_KIND;
```

```
typedef enum {  
    S_NULL,  
    S_AUTO,  
    S_STATIC,  
    S_TYPEDEF,  
    S_EXTERN,  
    S_REGISTER  
} S_KIND;
```

```
typedef enum {  
    ID_NULL,  
    ID_VAR,  
    ID_FUNC,  
    ID_PARM,  
    ID_FIELD,  
    ID_TYPE,  
    ID_ENUM,  
    ID_STRUCT,  
    ID_ENUM_LITERAL  
} ID_KIND;
```

```
typedef struct s_node {  
    NODE_NAME name;  
    int line;  
    int value;  
    struct s_type *type;  
    struct s_node *link;  
    struct s_node *clink;  
    struct s_node *rlink;  
} A_NODE;
```

```
typedef struct s_type {  
    T_KIND kind;  
    int size;  
    int local_var_size;  
    struct s_type *element_type;  
    struct s_id *field;  
    struct s_node *expr;  
    int line;  
    BOOLEAN check;  
    BOOLEAN prt;  
} A_TYPE;
```

```
typedef struct s_id {  
    char *name;  
    ID_KIND kind;  
    S_KIND specifier;  
    int level;  
    int address;  
    int value;  
    A_NODE *init;  
    A_TYPE *type;  
    int line;  
    struct s_id *prev;  
    struct s_id *link;  
} A_ID;
```

```
typedef union {  
    int i;  
    float f;  
    char c;  
    char *s;  
} LIT_VALUE;
```

```
typedef struct lit {  
    int addr;  
    A_TYPE *type;  
    LIT_VALUE value;  
} A_LITERAL;
```

```
typedef struct {  
    A_TYPE *type;  
    S_KIND stor;  
    int line;  
} A_SPECIFIER;
```

```

%{
#include "type.h"
extern A_NODE *root;
extern A_ID *current_id;
extern int current_level;
extern A_TYPE *int_type;
#include <stdlib.h>
#include <stdio.h>
int yywrap();
int yyerror(char *s);
int main();
extern int yylex();
}%

%start program

%token IDENTIFIER TYPE_IDENTIFIER FLOAT_CONSTANT INTEGER_CONSTANT
CHARACTER_CONSTANT STRING_LITERAL
%token PLUS PLUSPLUS MINUS MINUSMINUS BAR AMP BARBAR AMPAMP ARROW
%token SEMICOLON LSS GTR LEQ GEQ EQL NEQ DOTDOTDOT
%token LP RP LB RB LR RR PERIOD COMMA EXCL STAR SLASH PERCENT ASSIGN
COLON
%token AUTO_SYM STATIC_SYM TYPEDEF_SYM STRUCT_SYM ENUM_SYM
SIZEOF_SYM UNION_SYM
%token IF_SYM ELSE_SYM WHILE_SYM DO_SYM FOR_SYM CONTINUE_SYM
BREAK_SYM
%token RETURN_SYM SWITCH_SYM CASE_SYM DEFAULT_SYM

%%
program
    : translate_unit
    ;

translate_unit
    : external_declaration
    | translate_unit external_declaration
    ;

external_declaration
    : function_definition
    | declaration
    ;

function_definition
    : declaration_specifiers declarator compound_statement
    | declarator compound_statement
    ;

declaration_list_opt
    : // no option
    | declaration_list
    ;

declaration_list
    : declaration
    | declaration_list declaration
    ;

declaration
    : declaration_specifiers init_declarator_list_opt SEMICOLON
    ;

declaration_specifiers
    : type_specifier
    | storage_class_specifier
    | type_specifier declaration_specifiers
    | storage_class_specifier declaration_specifiers
    ;

storage_class_specifier
    : AUTO_SYM
    | STATIC_SYM
    | TYPEDEF_SYM
    ;

init_declarator_list_opt
    : // no option
    | init_declarator_list
    ;

```

```

init_declarator_list
    : init_declarator
    | init_declarator_list COMMA init_declarator
    ;

init_declarator
    : declarator
    | declarator ASSIGN initializer
    ;

initializer
    : constant_expression
    | LR initializer_list RR
    ;

initializer_list
    : initializer
    | initializer_list COMMA initializer
    ;

type_specifier
    : struct_type_specifier
    | enum_type_specifier
    | TYPE_IDENTIFIER
    ;

struct_type_specifier
    : struct_or_union IDENTIFIER LR struct_declaration_list RR
    | struct_or_union LR struct_declaration_list RR
    | struct_or_union IDENTIFIER
    ;

struct_or_union
    : STRUCT_SYM
    | UNION_SYM
    ;

struct_declaration_list
    : struct_declaration
    | struct_declaration_list struct_declaration
    ;

struct_declaration
    : type_specifier struct_declarator_list SEMICOLON
    ;

struct_declarator_list
    : struct_declarator
    | struct_declarator_list COMMA struct_declarator
    ;

struct_declarator
    : declarator
    ;

enum_type_specifier
    : ENUM_SYM IDENTIFIER LR enumerator_list RR
    | ENUM_SYM LR enumerator_list RR
    | ENUM_SYM IDENTIFIER
    ;

enumerator_list
    : enumerator
    | enumerator_list COMMA enumerator
    ;

enumerator
    : IDENTIFIER
    | IDENTIFIER ASSIGN expression
    ;

declarator
    : pointer direct_declarator
    | direct_declarator
    ;

pointer
    : STAR
    | STAR pointer
    ;

```

```

direct_declarator
    : IDENTIFIER
    | LP declarator RP
    | direct_declarator LB constant_expression_opt RB
    | direct_declarator LP parameter_type_list_opt RP
    ;

```

```

parameter_type_list_opt
    : // no option
    | parameter_type_list
    ;

```

```

parameter_type_list
    : parameter_list
    | parameter_list COMMA DOTDOTDOT
    ;

```

```

parameter_list
    : parameter_declaration
    | parameter_list COMMA parameter_declaration
    ;

```

```

parameter_declaration
    : declaration_specifiers declarator
    | declaration_specifiers abstract_declarator_opt
    ;

```

```

abstract_declarator_opt
    : // no option
    | abstract_declarator
    ;

```

```

abstract_declarator
    : direct_abstract_declarator
    | pointer
    | pointer direct_abstract_declarator
    ;

```

```

direct_abstract_declarator
    : LP abstract_declarator RP
    | LB constant_expression_opt RB
    | direct_abstract_declarator LB constant_expression_opt RB
    | LP parameter_type_list_opt RP
    | direct_abstract_declarator LP parameter_type_list_opt RP
    ;

```

```

statement_list_opt
    : // no option
    | statement_list
    ;

```

```

statement_list
    : statement
    | statement_list statement
    ;

```

```

statement
    : labeled_statement
    | compound_statement
    | expression_statement
    | selection_statement
    | iteration_statement
    | jump_statement
    ;

```

```

labeled_statement
    : CASE_SYM constant_expression COLON statement
    | DEFAULT_SYM COLON statement
    ;

```

```

compound_statement
    : LR declaration_list_opt statement_list_opt RR
    ;

```

```

expression_statement
    : SEMICOLON
    | expression SEMICOLON
    ;

```

```

selection_statement
    : IF_SYM LP expression RP statement
    | IF_SYM LP expression RP statement ELSE_SYM statement
    | SWITCH_SYM LP expression RP statement
    ;

```

```

iteration_statement
    : WHILE_SYM LP expression RP statement
    | DO_SYM statement WHILE_SYM LP expression RP SEMICOLON
    | FOR_SYM LP for_expression RP statement
    ;

```

```

for_expression
    : expression_opt SEMICOLON expression_opt SEMICOLON expression_opt

```

```

expression_opt
    : // no option
    | expression
    ;

```

```

jump_statement
    : RETURN_SYM expression_opt SEMICOLON
    | CONTINUE_SYM SEMICOLON
    | BREAK_SYM SEMICOLON
    ;

```

```

arg_expression_list_opt
    : // no option
    | arg_expression_list
    ;

```

```

arg_expression_list
    : assignment_expression
    | arg_expression_list COMMA assignment_expression
    ;

```

```

constant_expression_opt
    : // no option
    | constant_expression
    ;

```

```

constant_expression
    : expression
    ;

```

```

expression
    : comma_expression
    ;

```

```

comma_expression
    : assignment_expression
    ;

```

```

assignment_expression
    : conditional_expression
    | unary_expression ASSIGN assignment_expression
    ;

```

```

conditional_expression
    : logical_or_expression
    ;

```

```

logical_or_expression
    : logical_and_expression
    | logical_or_expression BARBAR logical_and_expression
    ;

```

```

logical_and_expression
    : bitwise_or_expression
    | logical_and_expression AMPAMP bitwise_or_expression
    ;

```

```

bitwise_or_expression
    : bitwise_xor_expression
    ;

```

```

bitwise_xor_expression
    : bitwise_and_expression
    ;

```

```

bitwise_and_expression
    : equality_expression
    ;

equality_expression
    : relational_expression
    | equality_expression EQL relational_expression
    | equality_expression NEQ relational_expression
    ;

relational_expression
    : shift_expression
    | relational_expression LSS additive_expression
    | relational_expression GTR additive_expression
    | relational_expression LEQ additive_expression
    | relational_expression GEQ additive_expression
    ;

shift_expression
    : additive_expression
    ;

additive_expression
    : multiplicative_expression
    | additive_expression PLUS multiplicative_expression
    | additive_expression MINUS multiplicative_expression
    ;

multiplicative_expression
    : cast_expression
    | multiplicative_expression STAR cast_expression
    | multiplicative_expression SLASH cast_expression
    | multiplicative_expression PERCENT cast_expression
    ;

cast_expression
    : unary_expression
    | LP type_name RP cast_expression
    ;

unary_expression
    : postfix_expression
    | PLUSPLUS unary_expression
    | MINUSMINUS unary_expression
    | AMP cast_expression
    | STAR cast_expression
    | EXCL cast_expression
    | MINUS cast_expression
    | PLUS cast_expression
    | SIZEOF_SYM unary_expression
    | SIZEOF_SYM LP type_name RP
    ;

postfix_expression
    : primary_expression
    | postfix_expression LB expression RB
    | postfix_expression LP arg_expression_list_opt RP
    | postfix_expression PERIOD IDENTIFIER
    | postfix_expression ARROW IDENTIFIER
    | postfix_expression PLUSPLUS
    | postfix_expression MINUSMINUS
    ;

primary_expression
    : IDENTIFIER
    | INTEGER_CONSTANT
    | FLOAT_CONSTANT
    | CHARACTER_CONSTANT
    | STRING_LITERAL
    | LP expression RP
    ;

type_name
    : declaration_specifiers abstract_declarator_opt
    ;

%%

```

```

int yywrap() { return(1);}

int yyerror(char *msg) { printf("%s \n",msg); exit(1);}

int main()
{
    yyparse();
    printf("Success\n");

    return 0;
}

```

### 부록 3. lex.l

```

digit      [0-9]
letter     [a-zA-Z_]
delim      [ \t]
line       [\n]
ws         {delim}+

%{
#include <string.h>
#include <stdio.h>
#include "y.tab.h"
#include "type.h"
extern A_ID *current_id;
int checkIdentifier(char *s);
}%

%%
{ws}      { }
{line}    { }
auto      { return(AUTO_SYM); }
break     { return(BREAK_SYM); }
case      { return(CASE_SYM); }
continue  { return(CONTINUE_SYM); }
default   { return(DEFAULT_SYM); }
do        { return(DO_SYM); }
else      { return(ELSE_SYM); }
enum      { return(ENUM_SYM); }
for       { return(FOR_SYM); }
if        { return(IF_SYM); }
return    { return(RETURN_SYM); }
sizeof    { return(SIZEOF_SYM); }
static    { return(STATIC_SYM); }
struct    { return(STRUCT_SYM); }
switch    { return(SWITCH_SYM); }
typedef   { return(TYPEDEF_SYM); }
union     { return(UNION_SYM); }
while     { return(WHILE_SYM); }
"++"      { return(PLUSPLUS); }
"--"      { return(MINUSMINUS); }
"->"      { return(ARROW); }
"<"       { return(LSS); }
">"       { return(GTR); }
"<="      { return(LEQ); }
">="      { return(GEQ); }
"=="      { return(EQL); }
"!="      { return(NEQ); }
"&&"      { return(AMPAMP); }
"||"      { return(BARBAR); }
"..."   { return(DOTDOTDOT); }
"("       { return(LP); }
")"       { return(RP); }
"["       { return(LB); }
"]"       { return(RB); }
"{"       { return(LR); }
"}"       { return(RR); }
":"       { return(COLON); }
"."       { return(PERIOD); }
","       { return(COMMA); }
"!"       { return(EXCL); }
"*"       { return(STAR); }
"/"       { return(SLASH); }
%"        { return(PERCENT); }
"&"       { return(AMP); }
";"       { return(SEMICOLON); }
"+"       { return(PLUS); }
"_"       { return(MINUS); }
"="       { return(ASSIGN); }
{digit}+  { return(INTEGER_CONSTANT); }
{digit}+\.{digit}+ { return(FLOAT_CONSTANT); }
{letter}({letter}{{digit}})* { return(checkIdentifier(yytext)); }
\"([^\n]|\\\"\\n)*\" { return(STRING_LITERAL); }
\\([^\n]|\\\"\\n)\\ { return(CHARACTER_CONSTANT); }
\"/\"[^\n]* { }
%%

```

```

int checkIdentifier(char *s) {
    char *table[] = {"int", "float", "char", "void"};

    for (int i = 0; i < 4; i++) {
        if (strcmp(table[i], s) == 0)
            return(TYPE_IDENTIFIER); // 예약된 키워드
    }

    return(IDENTIFIER); // 단순 이름
}

```

### 부록 4. hw4.sh

```

#!/bin/bash
yacc -d yacc.y
lex lex.l
cc y.tab.c lex.yy.c

DIR="./testdir"
for file in "$DIR"/*.c
do
    echo ""
    echo "Parsing $file..."
    ./a.out < "$file"
done

```

## 부록 5. 테스트 파일

### 1.c

```
void bubblesort(int *arr, int size) {
    int i, j;
    for (i = 0; i < size - 1; i++) {
        for (j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                int tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;
            }
        }
    }
}

int main() {
    int nicearray[6] = {10, 9, 8, 7, 6, 5};

    bubblesort(nicearray, sizeof(nicearray)/sizeof(int));

    return 0;
}
```

### 2.c

```
void bubblesort(int *arr, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                int tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;
            }
        }
    }
}

int main() {
    int nicearray[6] = {10, 9, 8, 7, 6, 5};

    bubblesort(nicearray, sizeof(nicearray)/sizeof(int));

    return 0;
}
```

### 3.c

```
main(){
    a;

    return 0;
}
```

### 4.c

```
int main(){
    int a;

    return 0;
}
```

### 5.c

```
int main() {
    int arr[5] = {[1] = 100, [3] = 100};

    return 0;
}
```

### 6.c

```
int main() {
    int arr[5] = {0, 100, 0, 100, 0};

    return 0;
}
```

### 7.c

```
void fun(int i){
    int x = i;
    return;
}

int main() {
    int x = 5;
    int y = x + 2;
    fun(x);

    return 0;
}
```

### 8.c

```
void fun(int i){
    int x = i;
    return;
}

int main() {
    int x = 5;
    fun(x);
    int y = x + 2;

    return 0;
}
```